Docket No. 50277-2248

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	Confirmation No.: 4456
Wei Li, et al	Group Art Unit: 2162
Serial No.: 10/643,629	Examiner: Shahid Al Alam
Filed: August 18, 2003	

For: FREQUENT ITEMSET COUNTING USING CLUSTERED PREFIXES AND INDEX SUPPORT

MS Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed on October 12, 2007. A Panel Decision from Pre-Appeal Brief Review was mailed December 5, 2007. The time period for filing this Appeal Brief is extended to January 7, 2008, as January 5, 2008, the one month date from the mailing of the Panel Decision, is a Saturday, and January 6, 2008, is a Sunday.

I. REAL PARTY IN INTEREST

Oracle International Corporation is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF CLAIMS

Claims 21-28 have been finally rejected and are the subjects of this appeal. Claims 29-34 are objected, but would otherwise be allowable if rewritten in independent form including all limitations of the base claim and any intervening claims.

IV. STATUS OF AMENDMENTS

The claims were not amended after the Final Office Action.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present application contains independent Claim 21, which is summarized below. Claims 23, 25, and 27 are argued separately from the independent Claim 21 upon which they depend, and therefore are summarized below.

The claims summarized below are annotated to cross-reference features of the claims to specific examples of those features disclosed in the specification. However, the annotations are not intended to limit the scope of the recited features to those specific examples to which the annotations refer.

Claim 21 recites (with added reference annotations in parenthesis) a method for performing a frequent itemset operation (par. [0021]), the method comprising the steps of: performing the frequent itemset operation in a plurality of phases (par. [0067]), wherein each phase is associated with combinations that have a particular number of items (par. [0067]); during at least one phase of the plurality of phases, performing the steps of determining candidate combinations that are to be evaluated during the phase (par. [0068]); grouping the candidate combinations into clusters (par. [0080]), wherein each cluster corresponds to a common combination of items (par. [0080]), and wherein all candidate combinations in a given cluster include the common combination of items associated with the cluster (par. [0080]); processing said candidate combinations, based on said clusters, to determine whether the candidate combinations satisfy a frequency criteria associated with said frequent itemset operation (par. [0087]); and storing, in a computer-readable medium, data that indicates which candidate combinations satisfy the frequency criteria associated with said frequent itemset operation (par. [0109]).

Claim 23 recites (with added reference annotations in parenthesis) a method similar to that of Claim 21, wherein the step of grouping the candidate combinations into clusters includes the step of establishing an ordering for said candidate combinations by sorting the candidate combinations relative to each other based on the items within each of the candidate combinations (par. [0083]).

Claim 25 recites (with added reference annotations in parenthesis) a method similar to that of Claim 23, wherein the step of processing the candidate combinations based on the clusters includes processing the candidate combinations in a sequence based on said ordering (par. [0083]).

Claim 27 recites (with added reference annotations in parenthesis) a method similar to that of Claim 21, wherein the step of grouping the candidate combinations into clusters includes hashing the candidate combinations into buckets based on the items that the candidate combination contain (par. [0084]).

Claim 29 recites (with added reference annotations in parenthesis) a method similar to that of Claim 21, wherein the step of processing the candidate combinations includes generating bitmaps for the candidate combinations, and determining how many item groups of an item group population include each candidate combination based on the bitmap for the candidate combination (par. [0086]).

Claim 31 recites (with added reference annotations in parenthesis) a method similar to that of Claim 29, wherein the step of processing the candidate combinations includes, for each cluster, performing the steps of: generating a bitmap for a particular combination that is a subcombination of all combinations in the cluster (par. [0086]); using the bitmap for the particular combination to generate bitmaps for all combinations in the cluster (par. [0086]); using the bitmap generated for each combination in the cluster to determine how many item groups include the combination (par. [0087]); and after all combinations in the cluster have been processed, discarding from volatile memory the bitmap for the particular combination (par. [0088]).

Claim 33 recites (with added reference annotations in parenthesis) a method similar to that of Claim 21, wherein the step of processing the candidate combinations includes generating bitmaps for the candidate combinations as the candidate combinations are processed in a sequence (par. [0090]), the method further comprising the steps of: generating one or more intermediary bitmaps for use in generating of a bitmap for a current candidate combination (par. [0091]); and after generating the bitmap for the current candidate combination, retaining in volatile memory only those intermediary bitmaps that are base bitmaps of a next candidate combination in said sequence (par. [0094]); and if any intermediate bitmaps are retained, then using one or more of the intermediary bitmaps to generate a bitmap for the next candidate combination in said sequence (par. [0095]).

Claims 22, 24, 26, 28, 30, 32, and **34** recite computer readable storage media (FIG. 6, storage device 610) that carry instructions for causing processors (FIG. 6, processor 604) to perform the steps of the methods of Claims 21, 23, 25, and 27, respectively.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 21-28 stand rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by U.S. Patent No. 6,324,533 ("*Agrawal*").

VII. ARGUMENTS

A. The Features of Claims 21 and 22 Are Not Anticipated by Agrawal

Claim 21 recites a method for performing a frequent itemset operation in phases. Claim 21 requires, among other things, "during at least one phase of the plurality of phases… *grouping the candidate combinations into clusters, wherein each cluster corresponds to a common combination of items,* and wherein all candidate combinations in a given cluster include the common combination of items associated with the cluster." The Examiner alleges that all limitations in Claim 21 are anticipated by *Agrawal*. While it is true that *Agrawal* describes an approach for performing a frequent itemset operation in phases, *Agrawal* lacks any teaching or suggestion of anything analogous to a cluster as claimed.

The Examiner cited *Agrawal*, Col. 5, lines 41-47, as anticipating the limitation of Claim 21 reciting grouping the candidate combinations into clusters. However, Col. 5, lines 41-47 merely states, *in toto*:

> The candidate generation procedure ensures that C_k is a superset of the set of all frequent k-itemsets. The algorithm builds a specialized hash-tree data structure in memory out of C_k. Data is then scanned in the support counting phase. For each transaction, the algorithm determines which of the candidates in C_k are contained in the transaction using the hash-tree data structure and increments their support count. At the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding F_k. The algorithm terminates when F_k or C_{k+1} becomes empty.

What is described in the cited passage of *Agrawal* is how candidate itemsets are counted during a pass or phase. Nothing is taught or suggested that even remotely may be interpreted as grouping the candidate combinations into clusters.

In addition, the Examiner states, in Office Action mailed July 12, 2007, p. 4:

> The frequent 1-itemsets and 2-itemsets are generated by referring a transaction table to obtain candidate set of (n+2)-itemsets and frequent(n+2)-itemsets using a query operation. The generation of frequency itemsets is repeated until the candidate set is empty. The mining rules are generated from the union of determine frequency itemsets. See abstract and C4:L57-67.

This describes how a phase or pass is generated, and not anything to do with grouping into clusters *after a set of candidate combinations has already* been determined. Rather, in *Agrawal*, candidate sets are generated based upon frequent itemsets of one less item.

In Claim 21, "each phase is associated with combinations that have a particular number of items." *Then*, once the candidate combinations within a phase are determined in Claim 21, "during at least one phase of the plurality of phases, performing the steps of determining candidate combinations that are to be evaluated during the phase; *grouping the candidate combinations into clusters*." *Agrawal* does *not* group any candidate combinations into clusters *after* "determining candidate combinations that are to be evaluated during the phase." Rather, it is apparent that *Agrawal* lacks this limitation. Both sections of *Agrawal* cited by the Examiner refer to candidate *generation*, and *not* grouping candidate combinations *after* the candidate combinations have already been generated.

To further draw the distinction of two separate steps performed, Applicants describe (1) eliminating candidates in phases using frequent itemsets (*Claim 21*, "each phase is associated with combinations that have a particular number of items"; and *Written Specification*, par. [0068]) and (2) once the candidates are in a phase, grouping the combinations into clusters (*Claim 21*, "during at least one phase…grouping the candidate combinations into clusters"; and *Written Specification*, par. [0080]). There is nothing analogous to *the second step* in *Agrawal.*

One of the reasons that the embodiments of the invention recited in Claims 21 and 22 are advantageous is because grouping candidate combinations into clusters reduces the amount of processing required to perform the evaluations of candidate combinations. For instance, if combinations are processed with bitmaps, "the problem of bitmap proliferation is reduced by (1) clustering the candidate combinations that are to be processed during the phase" (*Written Specification*, par. [0080]).

Therefore, the Examiner has *failed* to demonstrate that *Agrawal* discloses *all* of the limitations of Claim 21. Claim 22 recites a computer-readable storage medium that carries instructions for performing the steps of the method of Claim 21, including the features distinguished from *Agrawal*. As a result, Claims 21 and 22 are patentable over *Agrawal* under 35 U.S.C. § 102(b). The rejection of Claims 21 and 22 should be reversed.

B. Claims 23 and 24 Are Not Anticipated by Agrawal

By virtue of its dependence from Claim 21, Claim 23 includes the features of Claims 21 that are distinguished from *Agrawal.* Therefore, Claim 23 is patentable over *Agrawal.*

Additionally, Claim 23 recites the feature, "wherein the step of grouping the candidate combinations into clusters includes the step of establishing an ordering for said candidate combinations by sorting the candidate combinations relative to each other based on the items within each of the candidate combinations." The Examiner cites *Agrawal* column 6, line 67-column 7, line 1, as anticipating the claimed limitation. Specifically, the Examiner states that "a lexicographical ordering will ensure that the subsets are ordered by the item names included in the subset."

However, once the cited portion of *Agrawal* is placed into context, lexicographical ordering is performed *during the generating of the candidate combinations*, not during the *grouping of candidate combinations into clusters*. *Agrawal* states "the items in an itemset are *assumed* to be lexicographically ordered" (Agrawal, Col. 6, line 67- Col. 7, line 1), so that the candidate itemsets may be generated. Furthermore, the ordering is performed on the *frequent itemsets* to *generate* the candidate combinations, *not* on the candidate combinations themselves. Thus, not only is the ordering not performed on the analogous itemset, but the ordering is not performed during the same steps of the method either. As such, the Examiner has failed to show that *Agrawal* has anticipated each and every limitation as recited in Claim 23.

Claim 24 recites a computer-readable storage medium that carries instructions for performing the steps of the method of Claim 23, including the features distinguished from Agrawal.

As a result, Claims 23 and 24 are patentable over *Agrawal* under 35 U.S.C. § 102(b). The rejection of Claims 23 and 24 should be reversed.

C. <u>Claims 25 and 26 Are Not Anticipated by Agrawal</u>

By virtue of its dependence from Claim 23, Claim 25 includes the features of Claims 23 that are distinguished from *Agrawal*. Therefore, Claim 25 is patentable over Agrawal.

Additionally, Claim 25 recites the feature, "wherein the step of processing the candidate combinations based on the clusters includes processing the candidate combinations in a sequence based on said ordering." The Examiner alleges that this limitation is shown in *Agrawal* Col. 7, lines 4-14. The cited portions of *Agrawal* state:

> First, in the join step, a superset of the candidate itemsets C.sub.k is generated by joining F.sub.k-1 with itself, as shown in the following pseudo code:
> insert into C.sub.k select I.sub.1.item.sub.1, . . . , I.sub.1.item.sub.k-1, I.sub.2.item.sub.k-1
> from F.sub.k-1 I.sub.1, F.sub.k-1 I.sub.2
> where I.sub.1.item.sub.1 =I.sub.2.item.sub.1 and
> I.sub.1.item.sub.k-2 =I.sub.2.item.sub.k-2 and
> I.sub.1.item.sub.k-1 <I.sub.2.item.sub.k-1
> For example, let F.sub.3 be {{1 2 3}, {1 2 4}, {1 3 4}, {1 3 5}, {2 3 4}}. After the join step, C.sub.4 will be {{1 2 3 4}, {1 3 4 5}}.

Claim 25 recites "wherein the step of *processing the candidate combinations based on the clusters* includes processing." However, the cited portions of *Agrawal* describe generating the candidate items. *Agrawal* states "in the join step, a superset of the candidate itemsets C.sub.k is generated by joining F.sub.k-1 with itself." (*Agrawal* Col. 7, lines 4-5). *Agrawal* describes generating the candidate combinations themselves while Claim 25 describes steps *after the candidate combinations are generated* and placed in clusters. There is a non-sequitor by citing this portion of *Agrawal* because processing clearly may not occur on the candidate combinations until after the candidate combinations are already generated. Thus, it is not possible that the portion of *Agrawal* cited by the Examiner anticipates each and every limitation of Claim 25.

Claim 26 recites a computer-readable storage medium that carries instructions for performing the steps of the method of Claim 25, including the features distinguished from *Agrawal.*

As a result, Claims 25 and 26 are patentable over *Agrawal* under 35 U.S.C. § 102(b). The rejection of Claims 25 and 26 should be reversed.

D. Claims 27 and 28 Are Not Anticipated by Agrawal

By virtue of its dependence from Claim 21, Claim 27 includes the features of Claims 21 that are distinguished from *Agrawal.* Therefore, Claim 27 is patentable over *Agrawal.*

Additionally, Claim 27 recites the feature, "the step of grouping the candidate combinations into clusters includes hashing the candidate combinations into buckets based on the items that the candidate combination contain." The Examiner cites *Agrawal* Col. 12, lines 48-55 and Col. 13, lines 13-22 as anticipating this limitation.

The cited sections of *Agrawal* state:

> A table function Gather is used for creating the Tid-lists. This is the same as the Gather function in GatherJoin except here, the tid-list is created for each frequent item. The data table T is scanned in the (item, tid) order and passed to the function Gather. The function collects the tids of all tuples of T with the same item in memory and outputs a (item, tid-list) tuple for items that meet the minimum support criterion. The tid-lists are represented as BLOBs and stored in a new TidTable with attributes (item, tid-list).

```
insert into F.sub.k select item.sub.1, . . . , item.sub.k, count(tid-list) as cnt
        from (Subquery Q.sub.k) t where cnt>:minsup
Subquery Q.sub.l (for any l between 2 and k):
        select item.sub.1, . . . , item.sub.l,
                Intersect (r.sub.l-1.tid-list, t.sub.1.tid-list) as tid-list
```

```
from TidTable t.sub.1, (Subquery Q.sub.l-1) as r.sub.l-1,
        (select distinct item.sub.1, . . . item.sub.l from C.sub.k) as
        d.sub.l
where r.sub.l-1.item.sub.1 =d.sub.l.item.sub.1 and . . . and
        r.sub.l-1.item.sub.l-1 =d.sub.l.item.sub.l-1, and
        t.sub.l.item=d.sub.l.item.sub.l
```

The cited portions of *Agrawal* describe methods of how to count the support of an itemset. "The function collects the tids of all tuples of T with the same item in memory and outputs a (item, tid-list) tuple for items that meet the minimum support criterion." (*Agrawal*, col. 12, lines 52-54). However, nothing in the cited portions of *Agrawal* anticipate *hashing the candidate combinations into buckets.* Rather, *Agrawal* is transforming the data table into a vertical format by placing into a BLOB, all tids (transaction identification numbers) that contain the item and then counting the support by merging these tid-lists. (*Agrawal*, col. 12, lines 43-48). Claim 27, on the other hand, is hashing candidate combinations into buckets as one method to group the candidates into clusters. As *Agrawal* neither anticipates placing candidates into clusters, much less hashing candidate combinations into buckets to generate these clusters, the Examiner has *failed* to show that all limitations of Claim 27 are anticipated by *Agrawal.*

Claim 28 recites a computer-readable storage medium that carries instructions for performing the steps of the method of Claim 27, including the features distinguished from *Agrawal.*

As a result, Claims 27 and 28 are patentable over *Agrawal* under 35 U.S.C. § 102(b). The rejection of Claims 27 and 28 should be reversed.

E. Claims 29-34 Are Objected and Should be Allowed

Claims 29-34 are objected to as being dependant upon a rejected base claim, but would otherwise be allowable if rewritten in independent form including all limitations of the base claim and any intervening claims. Claims 30, 32, and 34 recite computer-readable storage mediums that carry instructions for performing the steps of the method of Claims 29, 31, and 33. As the Examiner has only objected to Claims 29-34, these claims are patentable and should be allowed.

CONCLUSION AND PRAYER FOR RELIEF

Based on the foregoing, it is respectfully submitted that the rejections of Claims 21-28 lack the requisite factual and legal bases. Appellants respectfully request that the Honorable Board reverse the rejections of Claims 21-28.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

/RobertSChee#58554/

Date: January 7, 2008

Robert S. Chee Registration No. 58,554

2055 Gateway Place, Suite 550
San Jose, California 95110-1083
Tel: (408) 414-1080
Fax: (408) 414-1076

VIII. CLAIMS APPENDIX

21. A method for performing a frequent itemset operation, the method comprising the steps of:

performing the frequent itemset operation in a plurality of phases, wherein each phase is associated with combinations that have a particular number of items;

during at least one phase of the plurality of phases, performing the steps of

determining candidate combinations that are to be evaluated during the phase;

grouping the candidate combinations into clusters, wherein each cluster corresponds to a common combination of items, and wherein all candidate combinations in a given cluster include the common combination of items associated with the cluster;

processing said candidate combinations, based on said clusters, to determine whether the candidate combinations satisfy a frequency criteria associated with said frequent itemset operation; and

storing, in a computer-readable medium, data that indicates which candidate combinations satisfy the frequency criteria associated with said frequent itemset operation.

22. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 21.

23. The method of Claim 21, wherein the step of grouping the candidate combinations into clusters includes the step of establishing an ordering for said candidate combinations by sorting the candidate combinations relative to each other based on the items within each of the candidate combinations.

24. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 23.

25. The method of Claim 23, wherein the step of processing the candidate combinations based on the clusters includes processing the candidate combinations in a sequence based on said ordering.

26. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 25.

27. The method of Claim 21, wherein the step of grouping the candidate combinations into clusters includes hashing the candidate combinations into buckets based on the items that the candidate combination contain.

28. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 27.

29. The method of Claim 21, wherein the step of processing the candidate combinations includes generating bitmaps for the candidate combinations, and determining how many item groups of an item group population include each candidate combination based on the bitmap for the candidate combination.

30. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 29.

31. The method of Claim 29, wherein the step of processing the candidate combinations includes, for each cluster, performing the steps of:

generating a bitmap for a particular combination that is a subcombination of all combinations in the cluster;

using the bitmap for the particular combination to generate bitmaps for all combinations in the cluster;

using the bitmap generated for each combination in the cluster to determine how many item groups include the combination; and

after all combinations in the cluster have been processed, discarding from volatile memory the bitmap for the particular combination.

32. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 31.

33. The method of Claim 21, wherein the step of processing the candidate combinations includes generating bitmaps for the candidate combinations as the candidate combinations are processed in a sequence, the method further comprising the steps of:

generating one or more intermediary bitmaps for use in generating of a bitmap for a current candidate combination; and

after generating the bitmap for the current candidate combination, retaining in volatile memory only those intermediary bitmaps that are base bitmaps of a next candidate combination in said sequence; and

if any intermediate bitmaps are retained, then using one or more of the intermediary bitmaps to generate a bitmap for the next candidate combination in said sequence.

34. A computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 33.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.